Joint Runtime / Energy Optimization and Hardware / Software Partitioning of Linear Transforms

Paolo D’Alberto, Franz Franchetti, Peter A. Milder, Aliaksei Sandryhaila, James C. Hoe, José M. F. Moura, Markus Püschel

Department of Electrical and Computer Engineering
Carnegie Mellon University

Jeremy R. Johnson
Department of Computer Science
Drexel University

Sponsors: DARPA DESA program, NSF-NGS/ITR, NSF-ACR, and Intel
The Problem

- The “Power Wall” is the new performance bottleneck
- Where to find power-efficient performance
  - multi-core, many-core, heterogeneous mutli-core
  - specialized HW computations
  - SW/HW systems
- Real SW/HW Platforms oday
  - Embedded: Xilinx Virtex Pro
    PowerPCs within FPGA
  - Desktop PC: XtremeData XD10000
    AMD Opterons and FPGAs in a hypertransport shared memory PC
  - Supercomputing: SGI RASC
    Itanium-based server + FPGA

*Must programmers become hardware designers?!*
SW/HW Partitioning in a Nutshell

Basic Idea

- Fixed set of compute-intensive primitives in HW
  → performance/power/energy efficiency
- Control-intensive SW on CPU
  → flexibility in functionality
- E.g. support a library of many DFT sizes efficiently

Manual Design Nightmare

- Decide which functionality to support in HW while co-optimizing HW and SW components
- Build HW modules
- Code SW library using the modules
- Optimize for performance/energy trade-off

Spiral
Automatic Performance Tuning

- BLAS: ATLAS/LAPACK (U. Tennessee)
- Linear algebra: Bebop (Berkeley), Flame (UT Austin)
- Quantum computations (Ohio State)
- Sorting (UIUC)
- Fourier transform: FFTW (MIT)
- Linear transforms: Spiral
- ...others
- New compiler techniques

Proceedings of the IEEE special issue, Feb. 2005
Organization

- Spiral overview
- SW/HW Partitioning in Spiral
- Results
- Concluding remarks
Spiral

- Generates programs from a problem specification for linear transforms (DFT, DCT, DWT, filters, ....)
- Optimizes at the algorithmic level and at the code level
- Different code types supported: scalar, fixed point, vector, parallel, GPU, Verilog, SW+HW
- Different performance metrics supported
  Runtime, power, energy, energy delay product, resource usage
- Goal 1: A flexible push-button program generation framework for an entire domain of algorithms
- Goal 2: With new architectures, update the tool rather than the individual programs in the library

How Spiral Works

Spiral:
Complete automation of the implementation and optimization task

Basic idea:
Declarative representation of algorithms

Rewriting systems to generate and optimize algorithms

Problem specification (transform)

Algorithm Generation

Algorithm Optimization

Implementation

Code Optimization

Compilation

Compiler Optimizations

Fast executable

Spiral

Search

controls

algorithm

C code

performance
What is a (Linear) Transform?

- Mathematically: Matrix-vector multiplication

\[ x \leftrightarrow y = T \cdot x \]

- Example: Discrete Fourier transform (DFT)

\[ \text{DFT}_n = \left[ e^{-2k\ell\pi i/n} \right]_{0 \leq k, \ell < n} \]
Transform Algorithms: Example 4-point FFT

Cooley/Tukey fast Fourier transform (FFT):

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & j & -1 & -j \\
1 & -1 & 1 & -1 \\
1 & -j & -1 & j \\
\end{bmatrix}
= \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
\end{bmatrix}
\]

Fourier transform

Diagonal matrix (twiddles)

\[
\text{DFT}_4 = (\text{DFT}_2 \otimes I_2) T_2^4 (I_2 \otimes \text{DFT}_2) L_2^4
\]

- Algorithms reduce arithmetic cost \( O(n^2) \rightarrow O(n \log(n)) \)
- Product of structured sparse matrices
- Mathematical notation exhibits structure: SPL (signal processing language)
Examples: Transforms

\[
\begin{align*}
\text{DCT-}2_n &= \left[ \cos(k(2\ell+1)\pi/2n) \right]_{0 \leq k, \ell < n}, \\
\text{DCT-}3_n &= \text{DCT-}2_n^T \quad \text{(transpose)}, \\
\text{DCT-}4_n &= \left[ \cos((2k+1)(2\ell+1)\pi/4n) \right]_{0 \leq k, \ell < n}, \\
\text{IMDCT}_n &= \left[ \cos((2k+1)(2\ell+1+n)\pi/4n) \right]_{0 \leq k < 2n, 0 \leq \ell < n}, \\
\text{RDFT}_n &= [r_{k\ell}]_{0 \leq k, \ell < n}, \quad r_{k\ell} = \begin{cases} 
\cos \frac{2\pi k\ell}{n}, & k \leq \left\lfloor \frac{n}{2} \right\rfloor \\
-\sin \frac{2\pi k\ell}{n}, & k > \left\lfloor \frac{n}{2} \right\rfloor 
\end{cases}, \\
\text{WHT}_n &= \begin{bmatrix} \text{WHT}_{n/2}^{n/2} & \text{WHT}_{n/2}^{n/2} \\ \text{WHT}_{n/2}^{n/2} & -\text{WHT}_{n/2}^{n/2} \end{bmatrix}, \quad \text{WHT}_2 = \text{DFT}_2, \\
\text{DHT} &= \left[ \cos(2k\ell\pi/n) + \sin(2k\ell\pi/n) \right]_{0 \leq k, \ell < n}.
\end{align*}
\]

Spiral currently contains 55 transforms
Examples: Breakdown Rules (currently \(\approx 220\))

\[
\begin{align*}
\text{DFT}_n & \rightarrow (\text{DFT}_k \otimes I_m) T^n_m(I_k \otimes \text{DFT}_m) L^n_k, \quad n = km \\
\text{DFT}_n & \rightarrow P_n(\text{DFT}_k \otimes \text{DFT}_m)Q_n, \quad n = km, \text{ gcd}(k, m) = 1 \\
\text{DFT}_p & \rightarrow R_p^T(I_1 \oplus \text{DFT}_{p-1}) D_p(I_1 \oplus \text{DFT}_{p-1}) R_p, \quad p \text{ prime} \\
\text{DCT-3}_n & \rightarrow (I_m \oplus J_m) L^n_m(\text{DCT-3}_m(1/4) \oplus \text{DCT-3}_m(3/4)) \\
& \quad \cdot (F_2 \otimes I_m) \begin{bmatrix} I_m & 0 & -J_{m-1} \\ \frac{1}{\sqrt{2}}(I_1 \oplus 2I_m) \end{bmatrix}, \quad n = 2m \\
\text{DCT-4}_n & \rightarrow S_n \text{DCT-2}_n \text{ diag}_{0 \leq k < n}(1/(2 \cos((2k + 1)\pi/4n))) \\
\text{IMDCT}_2m & \rightarrow (J_m \oplus I_m \oplus I_m \oplus J_m) \left( \begin{bmatrix} 1 \\ -1 \end{bmatrix} \otimes I_m \right) \oplus \left( \begin{bmatrix} -1 \\ -1 \end{bmatrix} \otimes I_m \right) J_{2m} \text{ DCT-4}_{2m} \\
\text{WHT}_2^k & \rightarrow \prod_{i=1}^{t} (I_{2^{k_1+k_2+\cdots+k_{i-1}}} \otimes \text{WHT}_{2^{k_i}} \otimes I_{2^{k_{i+1}+\cdots+k_t}}), \quad k = k_1 + \cdots + k_t \\
\text{DFT}_2 & \rightarrow F_2 \\
\text{DCT-2}_2 & \rightarrow \text{ diag}(1, 1/\sqrt{2}) F_2 \\
\text{DCT-4}_2 & \rightarrow J_2 R_{13\pi/8}
\end{align*}
\]

\[\text{Base case rules}\]
SPIRAL: Abstraction Levels

- **DFT<sub>256</sub>**
- **DFT<sub>16</sub>**

... (I4 ⊕ J4) D8(F2 ⊕ I4) ...

... \[ \sum_{j=0}^{n-1} S_{(j)m\otimes n} C_n G_{(j)m\otimes n} \]

- **transform**
- **ruletree**
- **SPL**
- **Σ-SPL**

- **SW (C/Fortran)**
- **SW vector/parallel**
- **SW/HW partitioned**
- **HW (RTL Verilog)**

- **machine code**
- **netlist**

- **Same infrastructure for SW, HW, SW/HW**
- **Optimization at the “right” level of abstraction**
- **Complete automation:** Conquers the high-level for automation
## SPL to Sequential Code

<table>
<thead>
<tr>
<th>SPL construct</th>
<th>code</th>
</tr>
</thead>
</table>
| \( y = (A_n B_n) x \) | \( t[0:1:n-1] = B(x[0:1:n-1]); \)
| \( y[0:1:n-1] = A(t[0:1:n-1]); \) |
| \( y = (I_m \otimes A_n) x \) | for \( i=0;i<m;i++ \)
| \( y[i*n:1:i*n+n-1] = \)
| \( A(x[i*n:1:i*n+n-1]) \) |
| \( y = (A_m \otimes I_n) x \) | for \( i=0;i<m;i++ \)
| \( y[i:n:i+m-1] = \)
| \( A(x[i:n:i+m-1]); \) |
| \( y = \left( \bigoplus_{i=0}^{m-1} A_n^i \right) x \) | for \( i=0;i<m;i++ \)
| \( y[i*n:1:i*n+n-1] = \)
| \( A(i, x[i*n:1:i*n+n-1]); \) |
| \( y = D_{m,n} x \) | for \( i=0;i<m*n;i++ \)
| \( y[i] = Dmn[i]*x[i]; \) |
| \( y = L_m^{mn} x \) | for \( i=0;i<m;i++ \)
| for \( j=0;j<n;j++ \)
| \( y[i+m*j] = x[n*i+j]; \) |

### Example: tensor product

\[ I_m \otimes A_n = \begin{bmatrix} A_n & \cdots & A_n \end{bmatrix} \]
SPL to Combinational Datapath (Verilog)

Primitives

\[ y = \text{DFT}_2 \cdot x, \quad \text{DFT}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \]

\[ y = L^4_2 \cdot x \]

\[ y = \text{diag}(d_0, d_1, d_2, d_3) \cdot x \]

Combination of primitives

\[ y = A_n B_n \cdot x \]

\[ y = (A_2 \otimes I_2) \cdot x \]

\[ y = (I_2 \otimes A_2) \cdot x \]
Organization

- Spiral overview
- SW/HW Partitioning in Spiral
- Results
- Concluding remarks
SPIRAL Approach

- **HW/SW partitioning**
  - Hardware (FPGA)
  - Shared Memory (Multicore)
  - Vector SIMD (SSE, VMX, …)
  - Message Passing (Clusters)
  - Graphics Processors (GPUs)
  - Multiple Levels of Parallelism (Cell BE)

_Spiral: One methodology optimizes for all types of parallelism_
Mixing HW and SW: The Math Perspective

- **Hardware:** regular computation
  
  \[ y = \left( I_p \otimes A \right) x \]

  streaming computation, full utilization

- **Software:** irregular computation + I/O
  
  \[ y = P \cdot x \]

  Task: maximize regularity under HW resource constraints

- SW: store HW output to main memory
HW/SW Partitioning Rewriting Rules

- SPL tag notation for codesign:
  $$A \text{ codesign}(s, DFT_{16})$$

  This means: formula $A$ is to be partitioned for streaming width $s$ and a $DFT_{16}$ HW block

- Rules transforms formulas into partitioned formulas
  - Formulas rewritten, tags propagated
  - There are choices

  $$\begin{align*}
  A \cdot B & \quad \rightarrow \quad A \text{ codesign}(s, A_m) \quad B \\
  I_m \otimes A_n \quad \text{codesign}(s, A_m, \ldots) & \quad \rightarrow \quad I_{m/s} \otimes (I_s \otimes A_n) \\
  A_m \otimes I_n \quad \text{codesign}(s, A_m, \ldots) & \quad \rightarrow \quad L_{mn}^{m/n} (I_{n/s} \otimes (I_s \otimes A_m)) \\
  A \quad \text{codesign}(s, A_m) & \quad \rightarrow \quad A \\
  \end{align*}$$

  functionality
  platform
  knowledge
  math knowledge
  with HW
  contextual info
HW/SW Partitioning by Rewriting

Partitioned *(hardware, software)* in the sense of our definition

- **Bulk** of computation done in hardware
- **I/O and unsupported computation** done in software
Optimizing a Whole Library

- **Goal:** Simultaneously optimize multiple problem sizes
  DFT library for N=16, 32, …, 8192

- **Problem:** Only a few cores fit on FPGA (area budget)
  Only resources for two DFT cores on FPGA

- **Additional Trick:** Virtual cores
  compute smaller DFT with larger core

Which cores to choose?
Choosing the Right Cores

- For all problem sizes and all choices of cores that fit on FPGA:
  - Partition and implement transform automatically
  - Measure implementation
- For each configuration:
  - Compute metric for whole library
- Pick best configuration w.r.t. metric

- Problem: Too slow (hardware synthesis in optimization loop)
- Answer: Search heuristic + hardware model
Organization

- Spiral overview
- SW/HW Partitioning in Spiral
- Results
- Concluding remarks
Xilinx XUP: our working environment

300 MHz

PPC 405

Cache

memory controller

to off-chip DDR DRAM

200 MB/sec OCM interface

~ 1M gates @ 100MHz

HW kernel in CLB logic

Xilinx Virtex-2 FPGA
Benchmark: Hardware-Only

- Automatically generated RTL Verilog for DFT 1024
- Flexible tradeoff: latency vs. area
- Beats vendor library from Xilinx
Benchmark: SW + 2 Cores (DFT\textsubscript{64} and DFT\textsubscript{256})

Latency performance

- DP search: Software only vs. SW + HW (both, DFT\textsubscript{512} only, DFT\textsubscript{64} only)
- For a library: 2 cores combine early ramp-up with high speed-up
Energy/Power: Which Cores to Choose?

Energy efficiency (latency mode)

- Software only vs. SW + 2 HW cores
- Clear winner for each size, but for whole library?
- What about performance/power trade-off?

Problem size

Energy efficiency [Mop/J]

- SW + DFT32 + DFT256
- SW + DFT64 + DFT512
- SW + DFT128 + DFT1024
- SW only
- 1733 MHz Athlon
Pareto: Power vs. Performance

- **Software only vs. SW + 2 HW cores**
- **Whole library: all sizes equally important**

- 3x variation
- 4x variation
Organization

- Spiral overview
- SW/HW Partitioning in Spiral
- Results
- Concluding remarks
Same Approach for Parallelization

**Message Passing:** [ISPA 06]

$$\text{DFT}_{mn} \rightarrow \left( \text{DFT}_m \otimes I_n \right) \text{T}^{mn}_{n \text{par}} \left( I_m \otimes \text{DFT}_n \right) \text{L}^{mn}_{m \text{par}}$$

$$\text{vec}(\nu)$$

$$\cdots$$

$$\left( I_p \otimes L^{mn/p}_{m \text{par}} \right) \left( L^{p^2 \otimes I_{mn/p^2}}_{p \text{par}} \left( I_q \otimes \left( I_{p/q} \otimes L^{n \otimes I_{m/p}}_{q \text{par}} \otimes I_{m/q} \right) \right) \left( I_q \otimes I_{n/q} \otimes \text{DFT}_m \right)$$

$$\text{vec}(\nu)$$

$$\cdots$$

$$\left( I_{mn/p} \otimes L^{2\nu}_{m \text{par}} \right) \left( \text{DFT}_m \otimes I_{n/v} \otimes I_{\nu} \right) \text{T}^{mn\nu}_{n \text{par}} \left( I_{m/\nu} \otimes \left( L^{2\nu \otimes I_{n/\nu}}_{\nu \text{par}} \otimes I_{m/\nu} \right) \right) \text{L}^{mn\nu}_{m \text{par}}$$

$$\text{vec}(\nu)$$

$$\cdots$$

$$\left( I_{mn/\nu} \otimes L^{2\nu}_{m \text{par}} \right) \left( L^{2\nu \otimes I_{n/\nu}}_{\nu \text{par}} \otimes I_{m/\nu} \right) \left( I_{m/\nu} \otimes \left( L^{2\nu \otimes I_{n/\nu}}_{\nu \text{par}} \otimes I_{m/\nu} \right) \right) \left( \text{DFT}_m \otimes I_{\nu} \right)$$

$$\text{vec}(\nu)$$

**SIMD Vectorization:** [VECPAR 06, …]

$$\text{vec}(\nu)$$

**Cg/OpenGL for GPUs:**

$$\text{DFT}_{mn} \rightarrow \left( \text{DFT}_m \otimes I_n \right) \text{T}^{mn}_{n \text{par}} \left( I_m \otimes \text{DFT}_n \right) \text{L}^{mn}_{m \text{par}}$$

$$\text{vec}(\nu)$$

$$\cdots$$

**Shared Memory and Multicore:** [SC 06]

$$\text{vec}(\nu)$$

$$\text{vec}(\nu)$$

$$\cdots$$

$$\left( I_{mn/p} \otimes L^{0\nu}_{m \text{par}} \right) \left( \text{DFT}_m \otimes I_{n/v} \otimes I_{\nu} \right) \text{T}^{mn\nu}_{n \text{par}} \left( I_{m/\nu} \otimes \left( L^{0\nu \otimes I_{n/\nu}}_{\nu \text{par}} \otimes I_{m/\nu} \right) \right) \text{L}^{mn\nu}_{m \text{par}}$$

$$\text{vec}(\nu)$$

$$\cdots$$

$$\left( I_{mn/\nu} \otimes L^{0\nu}_{m \text{par}} \right) \left( L^{0\nu \otimes I_{n/\nu}}_{\nu \text{par}} \otimes I_{m/\nu} \right) \left( I_{m/\nu} \otimes \left( L^{0\nu \otimes I_{n/\nu}}_{\nu \text{par}} \otimes I_{m/\nu} \right) \right) \left( \text{DFT}_m \otimes I_{\nu} \right)$$

$$\text{vec}(\nu)$$

**With Bonelli, Lorenz, Ueberhuber, TU Vienna**

**With Shen, TU Denmark**

**With Voronenko, CMU**
Conclusions

- Automatic generation of very fast and fastest numerical kernels is possible and desirable

- **High level language and approach**
  Algorithm generation, algorithm optimization

- Same approach for loop optimization, different forms of parallelism, SW and HW implementations, and partitioning

- **Constraint optimization under different metrics**
  performance, power, energy, accuracy
(Part of the) Spiral Team

www.spiral.net